Exercices - Semaine 4

Raphaël Nedellec

Exercice : Créer son premier package: olympicsWeather

L'objectif de cet exercice sera de créer un package nous permettant d'accéder en temps réel à des prévisions météo, en s'adressant à l'API publique https://api.open-meteo.com/v1/forecast. L'objectif du package est de pouvoir obtenir depuis R des prévisions météo pour les différents sites olympiques.

1. Installez les packages `devtools`, `httr2`, `jsonlite`.
2. Dans une nouvelle session R, créer un nouveau projet de package intitulé `olympicsWeather` en utilisant la fonction `create_package` de la librairie `usethis`. Pourquoi n'est-il pas nécessaire d'installer explicitement `usethis` ?

Avant de restituer les données à l'utilisateur, nous allons écrire une fonction pour récupérer les données depuis l'api d'open-meteo.

3. Créer un nouveau script R intitulé `get_weather_forecast` en utilisant la fonction `usethis::use_r()`.
4. Sans implémenter la fonction pour l'instant, écrivez une requête d'api en utilisant la librairie `httr2` que vous aurez préalablement installé.

    - l'url de l'api open-api est la suivante: https://api.open-meteo.com/v1/forecast
    - vous utiliserez les fonctions `request`, `req_url_query`, `req_perform`, `resp_body_json` pour passer la requête. Ces instructions peuvent se chaîner avec des pipes. La fonction `request` permet d'initialiser l'objet de request en passant l'url de l'api. La fonction `req_url_query` permet de passer différents paramètres à la requête, quand `req_perform` exécute la requête et récupère les résultats dans la session R. Enfin, la fonction `resp_body_json` extrait les données retournées en json du résultat de la requête. Vous pourrez utiliser la fonction `tibble::as_tibble()` pour convertir en `tibble` la table retournée.
    - pour la première requête, nous souhaitons obtenir la prévision météo aux coordonnées géographiques `c(48.85, 2.35)`

- nous souhaitons obtenir les informations de températures, de températures ressenties, de probabilité de précipitation, et de quantités de précipitations. Indice, vous devrez passer le paramètre suivant (entre autres) `hourly= c("temperature_2m", "apparent_temperature", "precipitation_probability", "precipitation")` à la fonction `req_url_query`.

5. Décrivez le résultat. Qu'avons nous obtenu suite à notre requête ? Quels paramètres souhaitons nous changer si nous voulons pouvoir récupérer les prévisions météo pour tous les sites des JOs ?

6. L'objectif est donc d'implémenter une fonction `get_weather_forecast` pour récupérer une table de prévisions météo à une coordonnées GPS donnée. Avant d'implémenter cette fonction accessible à l'utilisateur, nous souhaitons implémenter plusieurs fonctions internes.

7. Fonction `perform_request`:

   - la fonction perform request prend en entrée deux arguments: latitude et longitude
   - elle effectue la requête sur l'api publique ci-dessus, et retourne le contenu du body sous forme d'une tibble (exactement comme dans la question 4)

8. Fonction `unnest_response`:

   - les résultats obtenus après la requête n'étant pas bien formattés, il faut écrire une fonction pour transformer la forme de la tibble obtenue
   - la fonction en entrée prendre une tibble au format issu de la requête
   - et retournera les mêmes données au schema suivant:
     - date_heure: heure au tz UTC
     - tempature_celsius: données de température
     - temperature_ressentie_celsius: données de température ressentie
     - precipation_proba: probabilité de pluie
     - precipitation: precipitation en mm

9. Tests unitaires pour la fonction `unnest_response`:

   - créez un script de tests en utilisant la fonction `usethis::use_test("unnest_response")`
   - dans ce script, créez un jeu de données minimal pour tester le comportement de la fonction
   - testez la fonction, en proposant plusieurs tests unitaires. Exemple de tests unitaires:
     - testez que la fonction renvoie le bon nombre de lignes
     - testez que les valeurs de la colonne temperature correspondent aux valeur proposées en entrée
     - testez le nom des colonnes en sortie
     - testez le nombre de colonnes en sortie

10. Lors de la séance 1, nous avons développé un outil permettant de trouver les coordonnées GPS à partir d'un nom d'adresse (en utilisant le package `tidygeocoder` et la fonction `reverse_geocode`). Nous souhaitons que l'utilisateur puisse obtenir des prévisions météos à partir de:

    - coordonnées GPS, i.e un vecteur numérique de taille 2
    - un nom de site olympique ou une adresse. A partir du code de la séance 1, définissez une fonction `address_to_gps` convertissant une adresse en coordonnées gps sous la forme d'un vecteur numérique de taille 2.

Définir une fonction `get_forecast` générique, et deux implémentations `get_forecast.character` et `get_forecast.numeric`.

11. Implémentez une fonction interne `get_gps_coordinate` renvoyant des coordonnées x,y pour une adresse en utilisant les fonctions de la question 10 (ou de la semaine 1).
12. Implémentez la fonction `get_forecast.numeric`. Cette fonction prend en entrée un argument `xy`, contenant un vecteur numérique de coordonnées x,y (latitude, longitude) de taille 2. Si cette condition n'est pas vérifiée, alors la fonction devra déclarer une erreur. La fonction appelera les deux fonctions `perform_request` et `unnest_response` avant de retourner la tibble de résultat.
13. Implémentez la fonction `get_forecast.character`. Cette fonction prend en entrée un argument `address`, de type `character`, de taille 1. Si cette condition n'est pas vérifiée, alors la fonction devra déclarer une erreur. La fonction appelera les deux fonctions `address_to_gps` puis la fonction `get_forecast` en passant le résultat de l'appel à `address_to_gps`.
14. Documentez la fonction `get_forecast` en utilisant les balises Roxygen appropriées.
15. Mettez à jour le fichier DESCRIPTION. Assurez-vous d'avoir un NAMESPACE à jour en utilisant `devtools::document`.

Question Bonus

16. Quelle stratégie employeriez-vous pour fournir une sortie visuelle aux utilisateurs ? Essayez de proposer une fonction dans le package pour rendre visuelle la sortie du package.